**Kseniia Vlasova**

# VEHICLE DETECTION SYSTEM AND TRACKING AVAILABILITY IN A PARKING LOT

**Thesis**

**CENTRIA UNIVERSITY OF APPLIED SCIENCES**

**Information Technology**

**June 2018**

**ABSTRACT**


| Centria University of Applied Sciences | Date | Author |
|---|---|---|
| | June 2018 | Kseniia Vlasova |


**Degree programme**
Information Technology

**Name of thesis**
VEHICLE DETECTION SYSTEM AND TARCKING AVAILABILITY IN A PARKING LOT

| Instructor | Pages |
|---|---|
| Dr Grzegorz Szewczyk | 25 |

**Supervisor**
Dr Grzegorz Szewczyk


This thesis is dedicated to the implementation of machine vision algorithms for monitoring parking availability. The system allows to detect and track vehicles in the parking lot, which functionality can be further extended such as empty space detection and possibility of reservation. It tracks the availability based on an analysis of image frame from the outdoor camera and marking the vehicles in the occupied places.

The designed system harnesses the powerful modularity of the C++ language and the OpenCV machine vision framework. For the user interface the Qt creator was used.
The system was created using the 4.5.2 version of Qt creator and 3.4.1 version of OpenCv library. It was initially developed on Linux OS machine.



**Key words**
C++, OpenCV, HOG, machine vision, vehicle, parking lot, Qt creator

**CONCEPT DEFINITIONS**

| | |
|---|---|
| Qt creator | Cross-platform, complete integrated development environment |
| HOG | Histogram of oriented gradients |
| C++ | General purpose programming language |
| OpenCV | Open Source Computer Vision Library |
| SVM | Support Vector machine |
| IDE | Integrated development environment |

**ASTRACT**

**CONCEPT DEFINITIONS**

**CONTENTS**

**GRAPHES**



**TABLES**

# 1 INTRODUCTION

The image recognition technology develops rapidly in recent years. Cameras have become cheaper, smaller, and of higher quality than ever before. With this advance and computational technologies, the computer vision has gradually become an essential part of the everyday life. The growth of the number of vehicles in a city leads to the issue of limited number of parking lots available, with the help of computer vision it has become easier and faster for creating a special monitoring system which searches for free parking spaces in order to reduce the amount of time required for searching the vacant place. The common ways to monitor the status of the parking lots are using different types of sensors. However, the problems of motion detection and tracking arise. The solution is using computer vision for this purpose.

Currently, to monitor the state of the parking lots the network of sensors is used, which includes weigh-in-motion sensors which are embedded into the pavement. However, the cost of the individual sensors becomes a limiting factor for large parking spots with many spaces to monitor. In comparison to sensor-based parking spaces detection, which requires installation and maintenance of the networks of sensors, vision-based systems are more cost-effective and easy in the maintenance.

The aim of this thesis was to create a system which allows to detect and track vehicles in a parking lot with a help of computer vision and analysis of the digital imagines obtained from outdoor camera. The system tracks the availability of spaces based on the vehicle presence and updates the state of parking lots. The idea of the project is that the camera detects the available places using the computer vision and image processing and shows to the user the vacant parking lots, indicating them as a green area and the vehicle itself is marked with a red frame. As a result, the user is able to see the availability of parking place in real time. The thesis consists of the following chapters. Chapter 2 provides the software specifications of the system and the main design principles. Chapter 3 contains a background theoretical description of the system working principles. Chapter 4 shows the software implementation of the vehicle detection system using the technologies such as OpenCV library, C++, and Qt Creator IDE for user interface. Last chapter is devoted to testing results made after the implementation and the prospects of the future development.

# 2 SOFTWARE SPECIFICATION

Requirements engineering is an essential part of building the software. It includes the specification of services which are required from the system as well as defines the constraints on the system's operation and development. User requirements are clear statements, made using the natural language together with the diagrams which describes the services the system is expected to provide to the user. System requirements are more detailed description of the software system's functions, services, and operational constrains. The different types of requirements are used for better connection and understanding between different types of readers. (Sommerville 2010.)

In the following sections, at first the functional and non-functional requirements as a part of specific software system requirements will be discussed. Then the graph is presented, in order to give a better understanding of the system architecture.

## 2.1 Software System Requirements

Process of defining the software system requirements is included into the development part of the system. The main aim is to produce an entire system specification privatization of the requirements. It allows the developers to interpret the client's needs and expectations for the future software. It is done by specifying the set of features and functions that the system must have. (Bruegge and Dutoit 2004, 14-15, 204, 604.)

### 2.1.1 Functional Requirements

Function requirements describes an objective of the system by defining the main behavior of software at the specific conditions. They describe the abstract way what the system's functions should be. Functional requirements shall not include the technical details about the software. Non-functional requirements are responsible for it. (Sommerville 2010, 85.) Functional requirements for vehicle detection system is presented below in TABLE 1. Overall there are four requirements presented for the software.

TABLE 1. Functional requirements

| | **Description** |
|---|---|
| FUN-01 | The system shall have an ability to identify the vehicle |
| FUN-02 | The system shall have an ability to detect the vehicle |
| FUN-03 | The system shall have an ability to mark detected vehicle |
| FUN-04 | The user shall have an ability to view the status of the parking place |

### 2.1.2 Non-functional Requirements

Non-functional requirements are requirements which specify how the system performs a certain function of its own. It describes the system behavior from the technical point of view. Also, it outlines the limitation of the system's functionality. Often the non-functional requirements specify the system's quality attributes or characteristics. (Sommerville 2010, 87.)

There is a considerate amount of sub categories of non-functional requirements. The tables below present one of the most essential of its type: performance TABLE 2, reliability TABLE 3, supportability TABLE 4, and usability TABLE 5.

TABLE 2. Performance requirements

| | **Description** |
|---|---|
| PER-01 | The system must be designed so that background tasks can continue while the user performs foreground tasks. |
| PER-02 | The system shall be able to respond to the user with not more than 1 minute. |
| PER-03 | The system shall have an ability to receive a picture from the camera every 20 seconds. |

TABLE 3. Reliability requirements

| | **Description** |
|---|---|
| REL-01 | The system must give stability and possibility to reuse the results in the future. |
| REL-02 | The system must be able to tolerate and operate the errors. |
| REL-03 | The system must be working in a real time which allows the users to view 24/7. |

TABLE 4. Supportability requirements

| | **Description** |
|---|---|
| SUP-01 | The system shall be able to support various operation systems such as Linux and Windows. |
| SUP-02 | The system shall be able to be extended in the future. |
| SUP-03 | The system shall be able to be maintainable. |

TABLE 5. Usability requirements

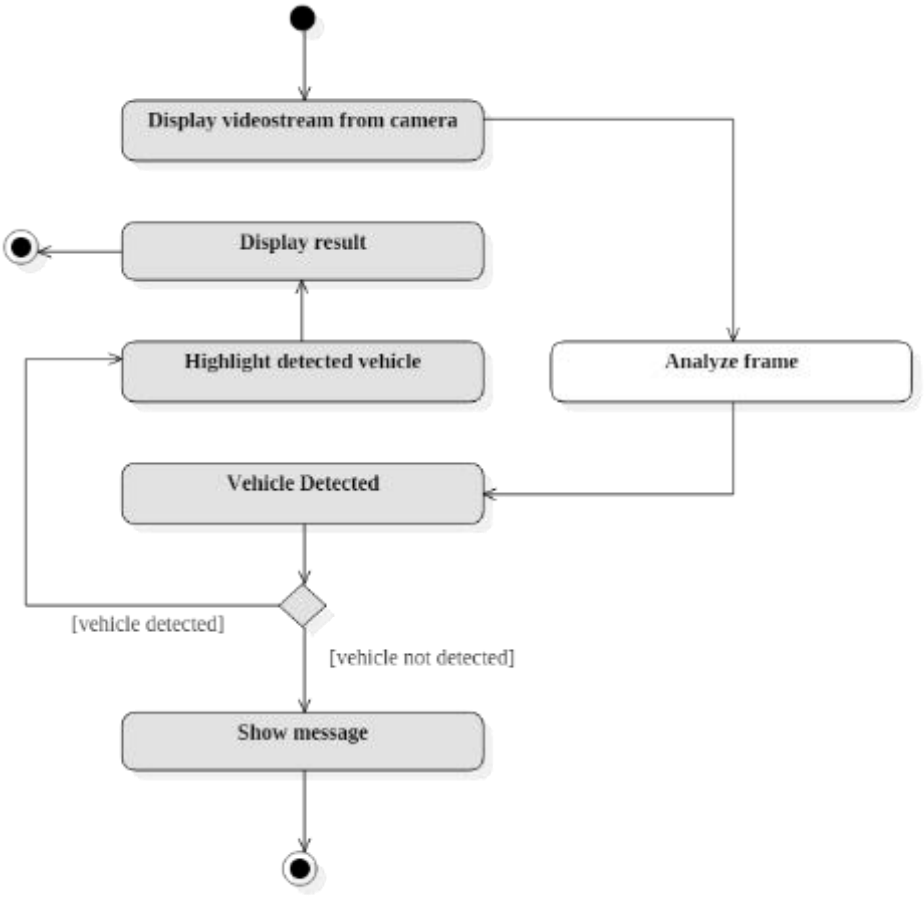
| | **Description** |
|---|---|
| USA-01 | The system must be easy to learn and does not distract the users' understanding of the system. |
| USA-02 | The system must guide users through an interface based on end concepts. |

### 2.1.3 Activity Diagram

The activity diagram is the object-oriented combination of flow charts and data-flow diagrams, which describes the workflow behavior of the system. It illustrates the dynamic flow of the system by modelling

the sequences of activities from one to another (Bruegge and Dutoit 2004). Diagram on GRAPH 1 shows the workflow of the vehicle detection and tracking system using the outdoor camera.

Firstly, the user can observe the status of the parking lot from the video. The vehicles themselves are marked with a read frame, determining that the place is occupied. At the same time, the places with no vehicle are marked as green areas, indicating the vacant status of the lot. In the meantime, the system captures the frames, received from the camera and analyzes them. Furthermore, the user interface provides the user the possibility to view the status of the parking lot. The system highlights the detected vehicles, as it was mentioned before. In the case, where no vehicles are detected, the system acknowledges the user with a popup message saying that the parking lot is fully available.



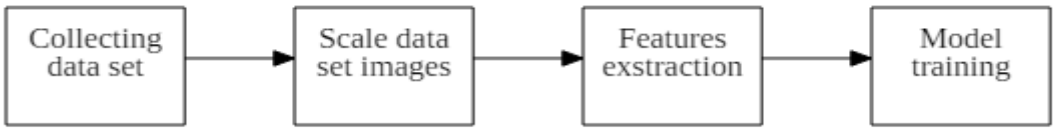

GRAPH 1. Activity diagram

# 3 THEORETICAL BACKGROUND

Vehicle detection system is a challenging problem. Nowadays a universal method does not exist. However, in order to achieve more discriminant detection, suggested to use the machine learning approach. The main idea is to automatically distinguish between classes of vehicle and non-vehicle. It can be achieved by presenting certain number of sets of the classes mentioned above to the system. (Ponsa, Serrat and Lo´pez 2011, 783–805.)

The vehicle detection system is based on supervised machine learning with a help of it, it is possible to recognize whether it is a vehicle or not. One of the most effective methods for vehicle recognition is histogram of oriented gradients(HOG) (Gepperth, Edelbrunner and Bucher 2005, 25–31).

Vehicle detection consists of four main steps which are presented in GRAPH 2. It shows the training part of the vehicle detection system which includes: a data set of samples, scaling the samples, computing the features and training a learning machine. The essential part of detection is highlighting the key feature space, which can uniquely identify the vehicle. After the set of features is fed to the classifier, the support vector machine (SVM) determines the class: vehicle or non-vehicle.




GRAPH 2. The training part of the vehicle detection system

## 3.1 Datasets

Datasets which are used for the machine leaning research in supervised learning set labels to the data. The vehicle detection system requires to have three datasets for more accurate results and decreasing the percentage of the error that might occur during the learning process. The total number of 4,374 samples in a dataset is divided into a training set and a testing set. The ratio between the training and the testing set might vary depending on the difficulty of the task. In the case of vehicle detection, the dataset was divided approximately equally between two sets.

### 3.1.1 Training dataset

The training dataset represents the set of training samples, or vectors of value. It is a collection of positive and negative data. Usually they are grouped by the same number of components or features. (Training Data 2014.)

The training dataset is an essential part of the future feature extraction needed for vehicle detection. The dataset which is used for this project is taken from two sources. The training set images are taken from the open source Car Dataset (Krause, Stark, Deng , & Fe, 2013, 554-561) and (Agarwal & Roth 2002, 113-127; Agarwal, Awan & Roth 2004).

GRAPH 3. Car image example from training set

It contains 2,165 images of cars from different angles and 2,209 images of cars, which represents the positive training set which is presented on GRAPH 3 and GRAPH 4 demonstrates the example of negative training set. Each of the image is scaled and resized to the size of 64 x 64 pixels.

GRAPH 4. Non-car image example from training set

### 3.1.2 Testing dataset

The testing dataset is defined as a data which provides the unprejudiced evaluation of the final model which is fit to the training dataset. It is the gold standard which is applied in practice after the model is

trained completely. The testing dataset contains various examples of the vehicle from different angles and scales. (Brownlee 2017.)

The testing dataset consists of 2,209 images 1,103 of which are single-scale, containing images of various sizes. However, the images of the cars are kept approximately in the same scale as in the training image, of which an example shown in GRAPH 5.

GRAPH 5. Test image single-scale

The rest of 1,106 images are of different sizes and contain cars of various scales and color, of which an example is presented in GRAPH 6.

GRAPH 6. Test images multi-scale

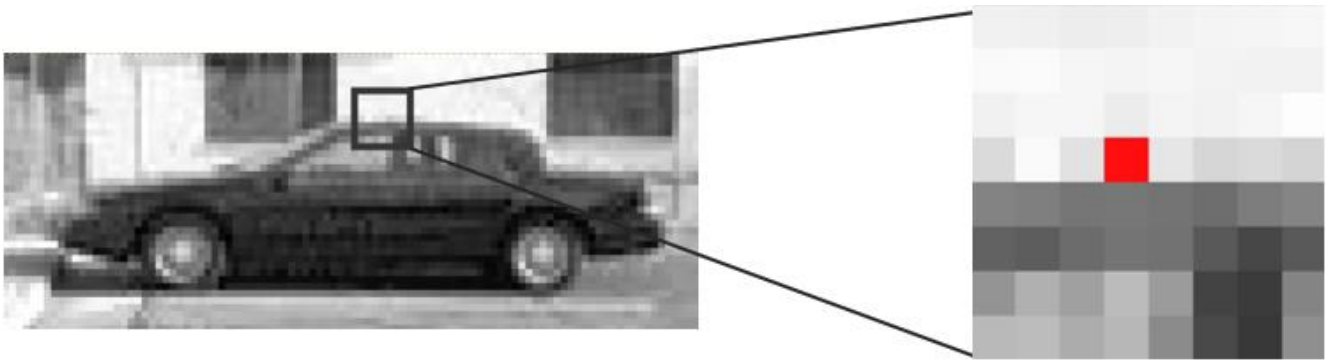
### 3.2 Features extraction with HOG

At this point the image includes too much information which is not needed for classification. The first step in feature extraction is extracting the most important information that contains in an image. The features that are extracted are the key features that can be recognizable in a vehicle. The most common features used in computer vision for vehicle detection is Histogram of Oriented Gradients or HOG. (Histogram of Oriented Gradients 2018.)

### 3.2.1 Gradient Histograms

Histogram of Oriented Gradients method was chosen for detection because it shows the high performance in car detection. It is a common method for shape recognition of an image (Kamenetsky and Sherrah 2015, 1-8). The main idea behind the HOG is that the algorithm converts an image of the fixed size to a feature vector of fixed size. The appearance and the shape can be described by the intensity distribution of gradients or direction of contours. (Kachouane, Sahki, Lakrouf, & Ouadah, 2004, 1475-1490.)

The implementation of HOG descriptor is computed by dividing the image into small connected regions called cells which size is 8x8, represented in GRAPH 7.



GRAPH 7. Gradient image computation

The pixel value varies from 0 to 255 (0 is black and 255 is white). The pixels values of the picture in GRAPH 7 are shown in TABLE 6.

TABLE 6. Pixel values

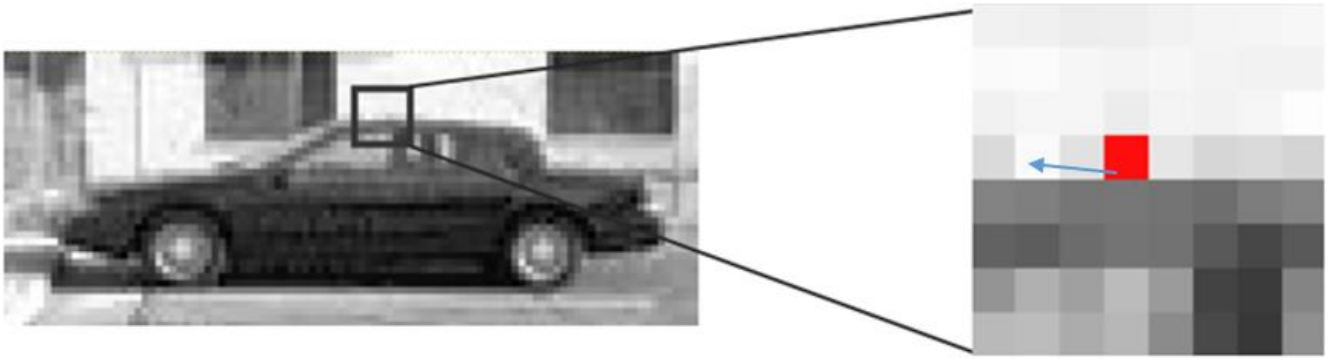
| | | |
|---|---|---|
| | 236 | |
| 226 | | 230 |
| | 119 | |

Furthermore, the rate of change in the x direction is 4 (230-226=4) and in y direction is 117 (236-119 = 117). As follows from the gradient vector which is x=4, y=117. It is possible to calculate the magnitude [1] and angle [2]:

$$Magnitude = \sqrt{x^2 + y^2} = \sqrt{4^2 + 117^2} = 117 \quad [1]$$

$$\text{Angle} = \tan^{-1}\left(\frac{x}{y}\right) = \tan^{-1}\left(\frac{4}{117}\right) = 1.95807° \quad [2]$$

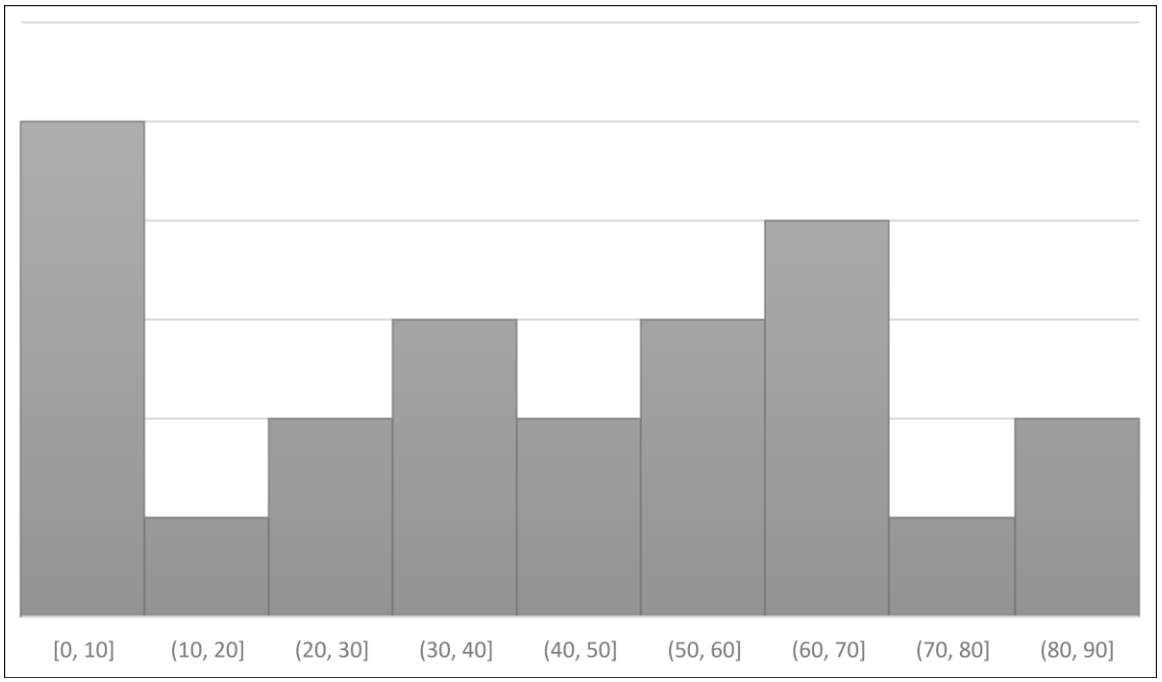
The result of the gradient computation is shown in GRAPH 8. After the mapping is performed to all 8x8 cells, pixels with a large negative change will be black and pixels with a large positive change will be white and pixels with no change will be gray.



GRAPH 8. Gradient vector

The GRAPH 9 of 9-bin histogram presents the contribution of all pixels in the 8x8 cells which are ajoined.

A histogram is an array of numbers where each element corresponds to the frequency of occurrence of a range of values for a set of data. In a part of an image, for instance, each box of the histogram may represent the pixels with the same color. A histogram is a transformation of the data space to the positive real numbers. The image is divided into cells of 8x8 pixels size and for each the histogram of oriented gradients is computed. (Kachouane, Sahki, Lakrouf, & Ouadah, 2004.)




GRAPH 9. Summary of the gradient computation

### 3.2.2 Histogram Normalization

Histogram normalization is done to avoid the change to the computed 8x8 pixel cell. In a case when the cell is multiplied by 1.5, for example, the magnitude of all gradients will be increased by 1.5. Histogram normalization prevents any illumination changed done to the calculated cells. (Histogram of Oriented Gradients 2018.)

Histogram normalization is performed by combining four cells into one block and calculating the normalized vector, by dividing the RGB color vector by the length of this vector. As a result, the normalized vector removes the scale, and there is the possibility now to perform changes to this vector.

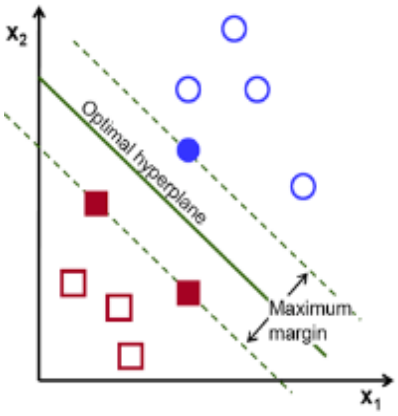
## 3.3 Training and classification

Classifier training is the most vital step of detecting the car, based on the feature set, for computer to learn to recognize whether the vehicle in an image or not. It is called classifying them as class labels, for example vehicle and non-vehicle. (OpenCV library 2018.)

There are two main steps to achieve the desired goal: training and classification itself. Training is the process of taking the known content, which belongs to specified classes, which results in creating a

classifier on this content. Classification is the next step which includes taking the training set and applying it to the unknown content. The output results depend on the quality of the training process, which is iterative while the classification is one-time process to run on unknown content. (Search Developer's Guide — Chapter 26 2018.)

Support Vector Machine (SVM) is one of the most popular supervised binary classification algorithm to perform classification whether it is a vehicle or not after being trained with prepared dataset. The GRAPH 10 shows the main idea lying behind the SVM. There are two classes exits (e.g. vehicles and non-vehicles) represented by two different types of dots. During the training the algorithm is provided with a vast number of examples from two classes. The algorithm's task is to separate those classes. The GRAPH 10 clearly shows the line between the two classes, which is called optimal hyperplane. The support vector machine will find the suitable plane that separates maximum of two classes. For training purposes, the 2,209 samples of vehicle images and 1,106 images of non-vehicle were used. The accuracy of feature accuracy is 0.9834.



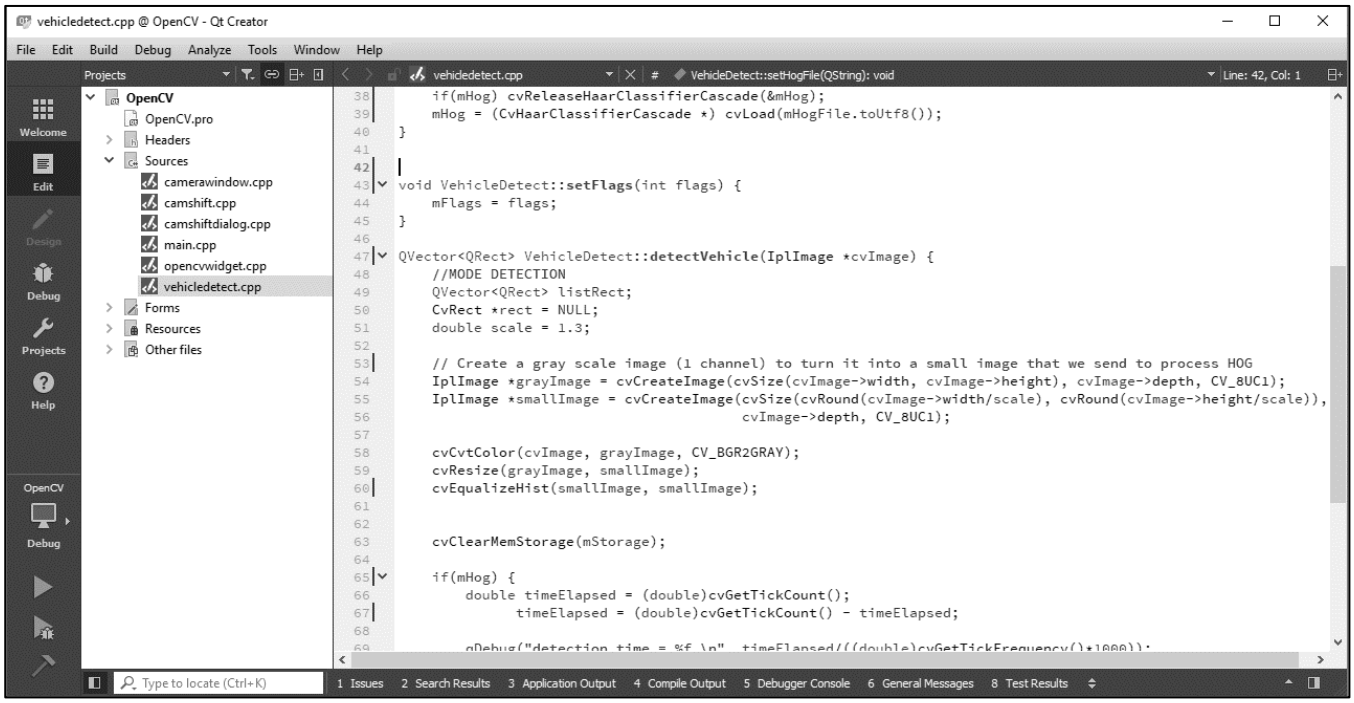

GRAPH 10. Support Vector Machine (SVM)

# 4 SOFTWARE IMPLEMENTATION

Developing the software is a time-consuming process which requires thoughtful planning of the design architecture and the result is achieved by try and error method. The vehicle detection system on a parking lot was implemented using the Qt Creator 4.5.2, OpenCV 3.4.1 library for vehicle recognition. The following chapter guides through the initial setup required for implementing the multiplatform software using the OpenCv library.

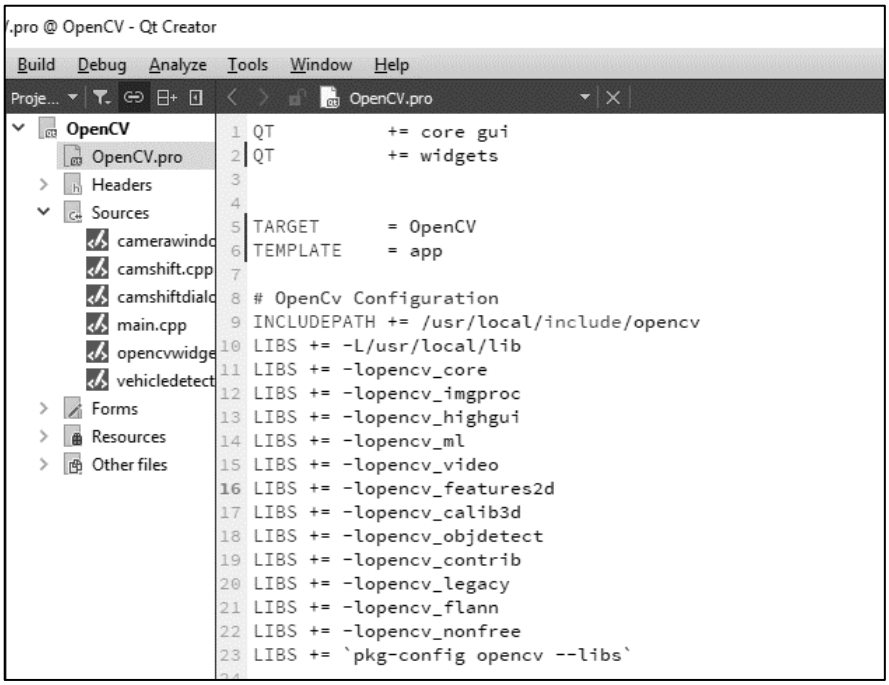
## 4.1 Qt Creator IDE

Qt Creator is a powerful, fully integrated development environment (IDE) which is provided with essential tools for designing and developing application with Qt application framework, on different platforms such as Linux, Windows and Android. GRAPH 11 shows the screenshot of the working environment (Qt Creator Manual 2018.) One of the reasons to use the Qt Creator is that there is a possibility to build on various platforms. For instance, the system was implemented initially on Linux OS Ubuntu 16.04 LTS machine and further built for Android phone version 7.0 and it was launched on Windows 10 OS machine successfully as well.




GRAPH 11. Qt Creator IDE

After the project is created it is important to export the OpenCv library into the project. It is done by editing the .pro file, which is generated automatically when the project is build. The OpenCv configuration lines are shown in GRAPH 12. It includes the essential libraries to work with video, image and object recognition.




GRAPH 12. QtCreator importing the OpenCv library

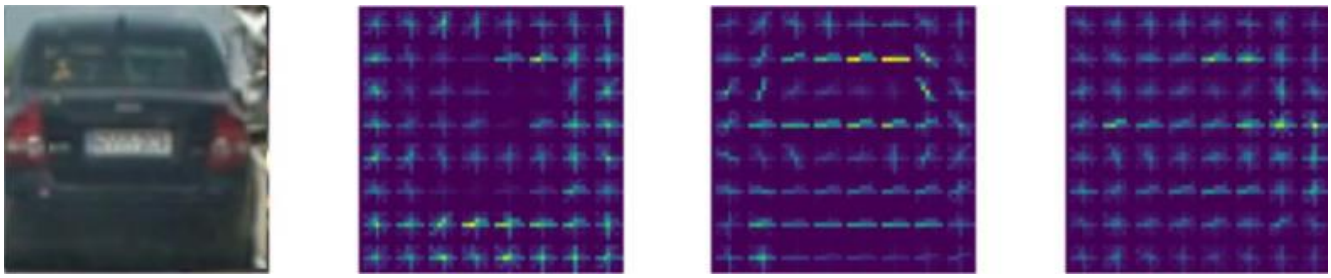
## 4.2 OpenCV library

OpenCv (Open Source Computer Vision Library) is a powerful open source software library. It consists of more than 2500 optimized computer vision and machine learning algorithms. The main purpose of using those algorithms is face detection and recognition, object identification, human actions classifications in videos, and camera movements tracking. The library is developed in C and C++ to enhance the computation efficiency, supported by main operation systems. (OpenCV library 2018.)

Vehicle detection system is created based on OpenCv library. The main purpose of using it is transforming the received frames, by analyzing them and making a representation of them. In order to implement the histogram of oriented gradients the gpu::HOGDescriptor class was used (Object Detection 2018). Class cvSVM::CVM was used to achieve the classification of two classes: vehicle and non-vehicles (Support Vector Machine 2018).

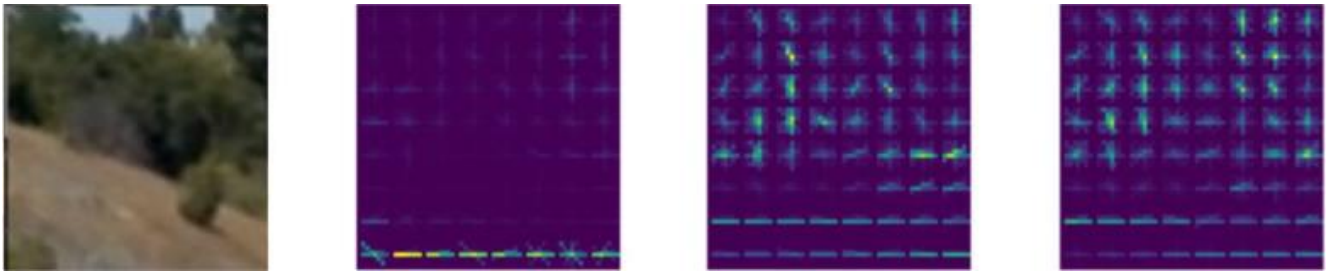
### 4.2.1 HOG descriptor

Histogram of Oriented gradients allows to identify the main features of the desired object to be detected. The results for HOG visualization for the vehicle are shown in GRAPH 13. The vectors and the directions form the main features of the vehicle, as it is clearly seen from the graph. The OpenCv library makes it easier to implement the HOG method. The special parameters entered specify the features, for instance the color model which is YUV, bounding box size, which indicates the size of the image, in this case it is 64x64 pixels.



GRAPH 13. HOG visualization for the vehicle

GRAPH 14 shows the result of the feature extraction from the non-vehicle object. It is clearly seen from he figures that the object has different shape compared to the first one. It does not show the main features of the vehicle, as a result it is classified as non-vehicle.



GRAPH 14. HOG visualization for non-vehicle

### 4.2.2 SVM classification

When the vehicle features were identified, the next step in vehicle detection is to teach the system to distinguish between two classes vehicle and non-vehicle. The OpenCv library provides the developer with essential tools for forming the hyperplane which is the key to separation of the two classes.

```
// Set up SVM's parameters
   CvSVMParams params;
   params.svm_type    = CvSVM::C_SVC;
   params.kernel_type = CvSVM::LINEAR;
   params.term_crit   = cvTermCriteria(CV_TERMCRIT_ITER, 100, 1e-6);

   // Train the SVM
   CvSVM SVM;
   SVM.train(trainingDataMat, labelsMat, Mat(), Mat(), params);

   Vec3b green(0,255,0), blue (255,0,0);


   // Show support vectors
   thickness = 2;
   lineType  = 8;
   int c     = SVM.get_support_vector_count();

   for (int i = 0; i < c; ++i)
   {
       const float* v = SVM.get_support_vector(i);
       circle( image,  Point( (int) v[0], (int) v[1]),   6,  Scalar(128, 128, 128), thickness, lineType);
   }

   imwrite("result.png", image);        // save the image


   waitKey(0);
}
```

GRAPH 15. Support Vector Machine implemented in QtCreator using OpenCv library

GRAPH 15 shows the main actions implemented to achieve the separation between two classes. First, the SVM's parameters are set, which is followed by training the SVM. Next, the results are shown to the user and the picture is saved as a 'result.png'.

# 5 TESTING

For the implementation and testing purposes the outdoor camera was installed near the parking lot of the block of flats. The load of the cars is not so vast, as a result the system was only tested successfully recognize one row of cars in a day time. The results show that due to the angel of the camera the system is not able to identify the vacant space between vehicle 1 and 2. GRAPH 16 represents the results of the parking lot, let view.

GRAPH 16. Parking lot view original picture on the left, the result of image analyzing on the right

The vehicle detection and tracking system was tested during the night time. However, due to the luck of enough light, the system was able to detect only the 24,6% of the vehicle presented on the parking lot. GRAPH 17 shows the same parking lot but from different angle. The results of detecting the vehicle from the back and straight were successful during the daytime only.

GRAPH 17. Parking lot view straight. The original picture is on the left and the result of image analyze is on the right.

## 5.1 Release notes

During the testing no serious issues have been found. The application is working as designed, according to the functional and non-functional requirements

## 5.2 Application further development

Vehicle detection system can have several features added in the future. Firstly, increase the detection rate during the night time in winter. Also, one of the future enhancements would be vehicle classification by size and by model. Finally, this technique can be applied to monitor other objects, such as closed parking lots.

**REFERENCES**

Agarwal, S., A. Awan, and D. Roth. 2004. "Learning to detect objects in images via a sparse, part-based representation." IEEE Transactions on Pattern Analysis and Machine Intelligence (IEEE) 26 (11): 1475-1490.

2002. Learning a Sparse Representation for Object Detection. Vol. 2353, in Computer Vision — ECCV 2002, by Shivani Agarwal and Dan Roth, edited by Peter Johansen, Anders Heyden, Gunnar Sparr and Ma, 113-127. Berlin, Heidelberg: Springer Berlin Heidelberg.

Brownlee, Jason. 2017. What is the Difference Between Test and Validation Datasets? July 14. Accessed 05 30, 2018. https://machinelearningmastery.com/difference-test-validation-datasets/.

Bruegge, Bernd, and Allen H. Dutoit. 2004. Object-Oriented Software Engineering: Using UML, Patterns and Java, 2nd Edition. Pearson.

Gepperth, A., J. Edelbrunner, and T. Bucher. 2005. "Real-time detection and classification of cars in video sequences." IEEE Intelligent Vehicles Symposium. Las Vegas, NV. 25–31.

2018. Histogram of Oriented Gradients. Accessed 05 30, 2018. https://www.learnopencv.com/histogram-of-oriented-gradients/.

Kachouane, M., S. Sahki, M. Lakrouf, and N. Ouadah. 2004. "HOG Based fast Human Detection." IEEE Transactions on Pattern Analysis and Machine Intelligence (IEEE) 26 (11): 1475-1490.

Kamenetsky, Dmitri, and Jamie Sherrah. 2015. "Aerial Car Detection and Urban Understanding." 1-8.

Krause, Jonathan, Michael Stark, Jia Deng , and Li Fe. 2013. "3D Object Representations for Fine-Grained Categorization." IEEE. 554-561.

2018. Object Detection. Accessed 30 05, 2018. https://docs.opencv.org/2.4/modules/gpu/doc/object_detection.html?highlight=hog.

2018. OpenCV library. Accessed 05 30, 2018. https://opencv.org/.

Ponsa, Daniel, Joan Serrat, and Antonio M. Lo´pez. 2011. "On-board image-based vehicle." Transactions of the Institute of Measurement and Control 783–805.

2018. Qt Creator Manual. Accessed 05 30, 2018. http://doc.qt.io/.

2018. Search Developer's Guide — Chapter 26. Accessed 05 30, 2018. https://docs.marklogic.com/guide/search-dev/classifier.

Sommerville, Ian. 2010. Software Engineering 9th. USA: Addison-Wesley Publishing Company.

2018. Support Vector Machine. Accessed 05 30, 2018. https://docs.opencv.org/2.4/modules/ml/doc/support_vector_machines.html#cvsvm.

2014. Training Data. Accessed 05 30, 2018. https://docs.opencv.org/3.0-beta/modules/ml/doc/mldata.html.